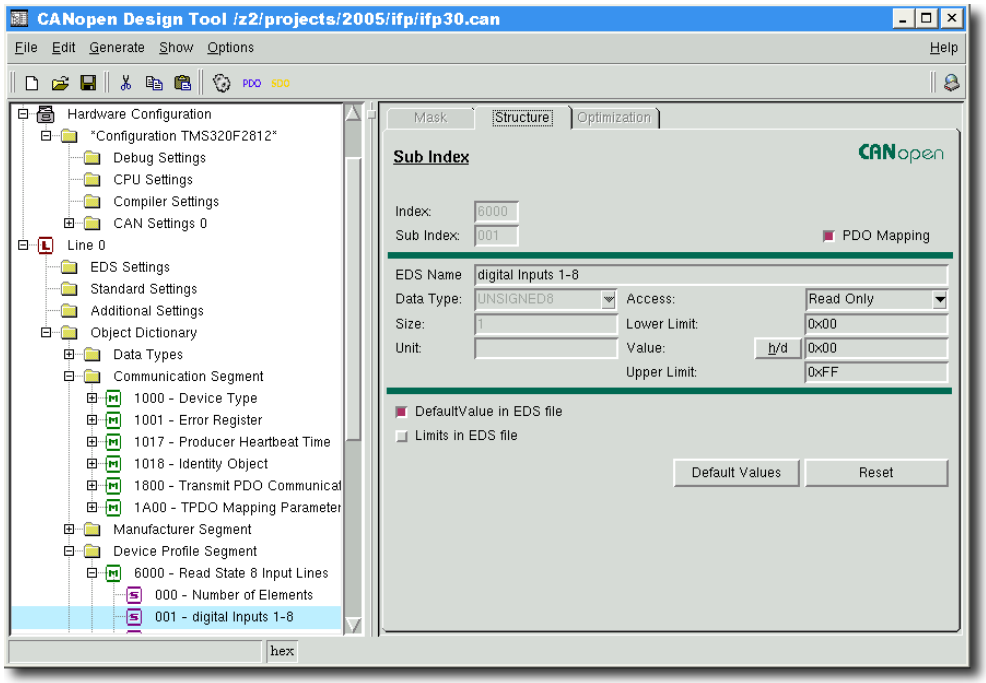
CANopen Design Tool
CANopen EDS Editor
Benutzerhandbuch



© *port* GmbH, Halle 13.01.2014; CANopen Design Tool Version 2.2.54

Ablehnungshinweis
Alle Rechte vorbehalten

Die von *port* GmbH gelieferten Programme, Baugruppen und Dokumentationen werden mit großer Sorgfalt erstellt und in unterschiedlichen Einsatzfällen getestet und geprüft.

port GmbH kann trotzdem keine Gewähr oder Haftung dafür übernehmen, dass die Software, die Baugruppe und die Dokumentation fehlerfrei bzw. für spezielle Zwecke geeignet ist.

Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

Für Folgeschäden, die aufgrund der Benutzung der Programme und Baugruppen auftreten, wird deshalb jede juristische Verantwortung oder Haftung ausgeschlossen.

port hat das Recht, Änderungen an den beschriebenen Produkten oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

Sämtliche Rechte an den Produkten einschließlich der Dokumentation liegen bei *port*. Die Weitergabe an Dritte und Vervielfältigung jeder Art, auch auszugsweise, sind nur mit schriftlicher Genehmigung durch *port* gestattet. Ausgenommen sind Arbeitskopien, die ausschließlich eigenen Zwecken dienen. Dabei trägt der Anwender die Verantwortung, dass die Kopien nicht in den Besitz Dritter gelangen.

Die in dieser Dokumentation verwendeten Soft- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen.

CANopen®
ist eine eingetragene Marke lizensiert durch CiA - CAN in Automation e.V, Deutschland.

EtherCAT®
ist eine eingetragene Marke und patentierte Technologie lizensiert durch die Beckhoff Automation GmbH, Deutschland.

Für Hinweise auf eventuelle Fehler sind wir dankbar und bitten um eine Benachrichtigung.

Wir werden uns bemühen, derartige Hinweise schnellstmöglich zu überprüfen.

Copyright

© 2014 *port* GmbH
Regensburger Straße 7
D-06132 Halle
Tel. +49 345 - 777 55 0
Fax. +49 345 - 777 55 20
E-Mail service@port.de
Internet http://www.port.de

Inhaltsverzeichnis

1. Einführung 7
1.1. Produktübersicht 7
1.2. Lieferumfang 7
1.3. Systemanforderungen 7
1.4. Installation 8
1.5. Unterstützung durch *port* 8

2. Aufbau des CANopen Design Tools 11
2.1. Hauptmenü 11
2.2. Objektbaum 12
2.2.1. Hardwarekonfigurationen 12
2.2.2. Objektverzeichnis 12
2.3. Aktionsmenüs 13
2.4. Eingabeformulare 14
2.4.1. Allgemeine Einstellungen 14
2.4.2. Allgemeine EDS/ESI/XDD-Einstellungen 15
2.4.3. Advanced configuration 15
2.4.4. Hardwarekonfigurationen 16
2.4.5. Linienspezifische EDS/XDD-Einstellungen 16
2.4.6. Linienspezifische Standardeinstellungen 16
2.4.7. Zusätzliche CANopen Einstellungen 16
2.4.8. Maskenansicht 16
2.4.9. Strukturansicht 17
2.4.10. Optimierungsansicht 19
2.5. Programmsteuerung 20

3. Dateistruktur 23
3.1. Projektstruktur 23
3.1.1. Projektdatei 23
3.1.2. Generierte Dateien 23

3.2. Profildateien 24
3.2.1. Profil CiA-443: SIIS level-2 devices 24

4. Projektbearbeitung 27
4.1. Projekterzeugung 27
4.2. Hardwarekonfiguration 27
4.3. Applikationsvariablen 28
4.3.1. Parametrierung von Applikationsvariablen 29
4.4. Kommunikationsvariablen 29
4.4.1. Parametrierung 30
4.5. Generierung des Quellcodes 30
4.6. %-Variablen 31
4.7. {}-Ausdrücke 31
4.8. Programmaufrufe nach der Generierung 32

5. Sonderfälle 33
5.1. Bereichsüberwachung von Variablenwerten 33
5.2. Gruppierung von Indizes in Subsegmenten 33
5.3. Herstellerspezifische Strukturen (Datentypen) 33
5.4. Vordefinierte Symbole 34
5.5. Spezielle Namen für Objektbeschreibungsstrukturen 34
5.6. Nicht-standardisierte Erweiterungen 34
5.7. Einschränkungen 34

6. Generierung einer anwenderspezifischen Objektbeschreibung 35
6.1. Formatbeschreiber 35
6.2. Beispiel für eine HTML Dokumentation 36
6.3. Beispiel für ein Tcl Script 37

7. CAN-Merge Erweiterungsmodul 39
7.1. Projektansicht 39
7.2. Detailansicht 40

8. CSV-Import Erweiterungsmodul 41

8.1. Anwendung 41
8.2. Beispiele 42

9. Unterschiedliche Versionen 43
9.1. Standardversion 43
9.2. Light Version 43
9.2.1. Einschränkungen 43
9.2.2. Anwendung 43
9.3. CANopen EDS Editor 44
9.3.1. Einschränkungen 44
9.3.2. Anwendung 44

10. Konvertierung bestehender Projekte 45
10.1. Bestehende Projekte - Version >= 2.2 45
10.2. Bestehende Projekte - Version >= 2.0 45
10.3. Bestehende Projekte - Version < 2.0 45
10.3.1. Das Konvertierungstool 46
10.3.2. Einschränkungen 47
10.3.2.1. Segmentierung des Objektverzeichnisses 47
10.3.2.2. Benutzerdefinierte Datentypen 47
10.4. Konvertierungssupport 47

1. Einführung

1.1. Produktübersicht

Das CANopen Design Tool von *port* ist ein Werkzeug zur Erstellung des Objektverzeichnisses für CANopen Applikationen. Es verwaltet Gerätedatenbanken, aus denen ein Objektverzeichnis, Konfigurations- und Initialisierungsdateien in C-Code, ein Electronic Data Sheet, eine XML Beschreibung und eine HTML-Dokumentation automatisch erzeugt werden. Weiterhin wird die Konfiguration der Treiberpakete durch das CANopen Design Tool erleichtert, wobei innerhalb eines Projekts mehrere Hardwarekonfigurationen verwaltet werden können.
Mit dem CANopen Design Tool steht ein Werkzeug zur Verfügung, welches den Entwickler von fehlerträchtigen und sich wiederholenden Tätigkeiten befreit. Es sichert die Konsistenz von implementierter Funktionalität , Electronic Data Sheet (EDS) und Gerätedokumentation. Im Lieferumfang sind Datenbanken mit dem CANopen-Kommunikationsprofil CiA-301 und CiA-302 enthalten. Optional sind Datenbanken zum Framework für sicherheitsrelevante Kommunikationsprofile CiA-304 verfügbar. Diese können durch Datenbanken zu verschiedenen Geräteprofilen, z.B. für E/A-Geräte nach CiA-401 oder Antriebe nach CiA-402 erweitert werden.
Das erzeugte Objektverzeichnis unterstützt zahlreiche Optionen der CANopen Library von *port*. Eine Baumdarstellung aller implementierten Parameter und Daten erleichtert die Wartung der Gerätesoftware. Mit dem CANopen Design Tool wird der Einstieg in das CANopen Protokoll erleichtert und die Geräteentwicklung beschleunigt.

1.2. Lieferumfang

Der Lieferumfang des CANopen Design Tools beinhaltet:

- Benutzerhandbuch
- Software für das CANopen Design Tool
- Datenbanken für Geräteprofile (optional)

Alle Komponenten stehen zum Download bereit.

Dieses Handbuch ist auch für den CANopen EDS Editor gültig. Für Anwender des CANopen EDS Editor sind einige im Handbuch beschriebene Eingabemasken und Ausgabedateien nicht verfügbar. In jedem Abschnitt wird separat darauf eingegangen, ob die Informationen nur für das CANopen Design Tool oder auch für den CANopen EDS Editor relevant sind.

1.3. Systemanforderungen

Das CANopen Design Tool ist ausführbar auf PC's mit Microsoft Windows™ oder Linux.

Betriebssystem: Windows 2000, Windows XP, Vista, Windows 7, Linux

Prozessor:	Pentium IV oder besser
RAM:	512 MByte
Festplattenspeicher:	45 MByte

1.4. Installation

Die Installation erfolgt menügesteuert über eine Setup-Datei. Entpacken Sie dafür die heruntergeladene ZIP-Datei und starten Sie **setup.exe** unter Windows™ bzw. setup.sh unter Linux. Nach der Installation finden sie die in Abbildung 1 dargestellte Verzeichnisstruktur wieder.

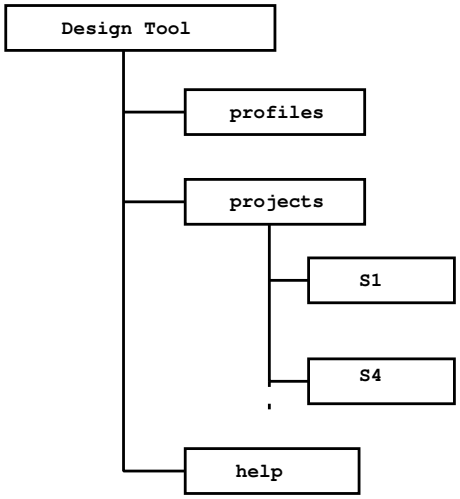


Abbildung 1, Verzeichnisstruktur des CANopen Design Tools

Unter **projects** befinden sich mitgelieferte Beispielprojekte. Für jedes Projekt ist nach Möglichkeit ein eigenes Verzeichnis zu verwenden. Unter **profiles** befinden sich die Datenbanken für die Kommunikationsprofile DS 301 , DS 302 und die CANopen-Geräteprofile. In diesem Verzeichnis können auch eigene Profile erstellt werden.
Das Verzeichnis **help** dient der Speicherung der Hilfedateien.

Nach der Installation auf einem Windows™-System kann das CANopen Design Tool über das Symbol auf dem Desktop oder über das Startmenü gestartet werden.

1.5. Unterstützung durch *port*

Die Ingenieure von *port* unterstützen die Anwender mit einer telefonischen Hotline und durch Schulungen. Weiterhin bietet *port* Beratungen im Umfeld von CAN, wie z.B. Netzwerkplanung, Netzwerkkonfiguration, Nachrichtenverteilung, Geräteauswahl und CANopen-Geräteprofilimplementierungen an.

Nähere Informationen erhalten Sie über

E-mail: **service@port.de**

Telefon: **+49 345 777 55 - 0**

Fax: **+49 345 777 55 - 20**

Die Ingenieure von *port* beantworten dann Ihre Nachricht schnellstmöglich.

2. Aufbau des CANopen Design Tools

Das CANopen Design Tool wird über Menüs, Werkzeugleisten, Baumansichten und Masken gesteuert.

2.1. Hauptmenü

Das Hauptmenü besteht aus den Einträgen:

File — Über den Menüpunkt *File* werden die Projekte erzeugt, geladen und geschlossen. Weiterhin kann das Programm in diesem Menü beendet werden.

Edit — Die Menüeinträge des Menüpunktes *Edit* dienen dem Kopieren, Löschen und Einfügen von Objekten im Objektverzeichnis. Sie haben keine Auswirkungen auf Werte in Eingabefeldern.

Generate — Bei Aktivierung des Eintrags *Generate all Files* im Menü *Generate* werden das Objektverzeichnis , die elektronischen Datenbeschreibungen , die Initialisierungsdatei **co_init.c** und die Konfigurationsdatei **cal_conf.h,** erzeugt. Ist zusätzlich *Generate Documentation* aktiviert, so wird ebenfalls die HTML-und Textdokumentation erstellt.

Show — Über den Menüpunkt *Show* lassen sich die erzeugten Dateien ansehen. Die HTML Dokumentation wird im Standard-Browser angezeigt.

Options — Unter *Options* sind die Optionen zur Erzeugung elektonischer Gerätebeschreibungen , die Darstellung des Baumes mit EDS-Namen oder Variablennamen , Auswahl spezieller Namen oder Strukturen für die Objektbeschreibungsstrukturen, und Optionen zur Einstellung des Verhaltens bei der Auswahl eines Objektes zusammengefasst. Weiterhin lassen sich an dieser Stelle die verwendeten Schriftarten und -größen konfigurieren. Optionen für die Generierung werden in der Projektdatei gespeichert. Dagegen werden Anzeige-Optionen benutzer-spezifisch in der Registry (oder in einer .rc-Datei unter Linux) gespeichert.

Help — Unterhalb von *Help* kann via *Context Help* eine Hilfe zu der aktuellen Ansicht und via *Manual* das Handbuch betrachtet werden. Die Hilfe wird im Standard-Browser angezeigt.

Abbildung 2, Werkzeugleiste

Die Werkzeugleiste unterhalb des Menüs bietet einen schnellen Zugriff auf die Funktionen: Neues Projekt, Öffnen, Speichern, Ausschneiden, Kopieren, Einfügen, Generieren, neue Dienste anlegen and Updates abfragen.

2.2. Objektbaum

Der Objektbaum ist zentrales Verwaltungselement für Geräteparameter, Hardwarekonfigurationen und alle CANopen Objekte. Er besteht aus Zweigen für die globalen Parameter, die Hardwarekonfigurationen und aus erweiterten Einstellungen den Zweigen für die zu unterstützenden Linien[1]. Die globalen ESI/EDS- bzw. XDD-Parameter sind Parameter wie z.B. Herstellername und Produktbezeichnung, welche für jede CAN-Linie gleich sind. Ebenfalls sind gelten die allgemeinen Einstellung und die allgemeinen Optimierungen für das gesamte Gerät. Mit diesen Einstellungen werden #define-Anweisungen für die Kompilierung der Bibliothek gesetzt. Weitere Informationen bietet die Kontexthilfe und das Handbuch der CANopen-Bibliothek.

Die einzelnen Linieneinträge teilen sich ihrerseits auf in:

- EDS/XDD[2] Einstellungen
- Standardeinstellungen (Linientyp und Optimierung)
- Zusätzliche CANopen Einstellungen
- Objektverzeichnis

2.2.1. Hardwarekonfigurationen

Innerhalb des CANopen Design Tools können mehrere Hardwarekonfigurationen verwaltet werden.
Eine Hardwarekonfiguration besteht aus:

- Debug Settings
  Einstellungen zu Debug-Nachrichten
- CPU Settings
  Auswahl und Konfiguration der verwendeten CPU oder des Betriebssystems
- Compiler Settings
  Auswahl und Konfiguration des verwendeten Compilers
- CAN Settings
  Auswahl und Konfiguration der verwendeten CAN-Controller oder der verwendeten CAN-Interface-Karte (eine CAN-Konfiguration pro CAN-Linie)

Mit den Einstellungen zur Hardwarekonfiguration werden Defines in der Datei **cal_conf.h** gesetzt, welche den jeweiligen Treiber konfigurieren.

2.2.2. Objektverzeichnis

Das Objektverzeichnis ist in verschiedene Segmente gegliedert.

- Datentypen (Indizes $0001_h - 025F_h$)
- Kommunikationsdaten (Indizes $1000_h - 1FFF_h$)
- herstellerspezifische Daten (Indizes $2000_h - 5FFF_h$)

[1] Ein Standard-CANopen Gerät unterstützt nur eine CAN-Linie.
Die Mehrlinienoption ist eine Besonderheit der Firma *port*
zur Realisierung von Gateways und Mehrliniensteuerungen.

[2] XDD ... XML Device Description (EDS-Nachfolgeformat)

- standardisierte Geräteprofilsegmentdaten (Indizes $6000_h - 9FFF_h$ in 800_h Blöcken)
- Bereich für DS 405-Geräte (Indizes $A000_h - AFFF_h$)
- reservierter Bereich $B000_h - FFFF_h$

Die Aufteilung des Bereiches für standardisierte Geräteprofile in 800_h Blöcke wurde unter besonderer Berücksichtigung des Multi-Device Profils realisiert, um in einem Gerät mehrere Profile implementieren zu können. Durch die Darstellung eines jeden Profils als eigenes Segment lassen sich Operationen, wie das Verdoppeln identischer Segmente, einfach realisieren. Weiterhin ist die Anordnung identischer Segmente innerhalb des Objektverzeichnisses mittels einer C-Struktur durch die Zuweisung eines gemeinsamen Variablennamens möglich. Das hat besondere Vorteile für den Variablenzugriff innerhalb des Gerätes z.B. für Mehrachsantriebe beim Wechseln zur anderen Achse mittels Feldindex. Diese Segmentierung ist ebenfalls für die herstellerspezifischen Daten möglich. Bei diesen kann der Start des 1. Blockes sowie die Größe der Blöcke frei parametriert werden. Eine Gruppierung von herstellerspezifischen und geräteprofilspezifischen Blöcken ist möglich, wenn diese eine identische Struktur besitzen.

Die einzelnen Zweige des Objektverzeichnisses beinhalten die CANopen Objekte. Diese Objekte gliedern sich wiederum in ihre Sub-Index-Elemente.

Bei Aktivierung der einzelnen Komponenten im Baum werden im rechten Programmfenster Aktionsmenüs z.B. für den Datenimport oder die jeweiligen Objektformulare angezeigt.

2.3. Aktionsmenüs

Die Aktionsmenüs unterscheiden sich für jeden Zweig des Objektbaumes. Es sind Menüs verfügbar für:

- Operationen für die Hardware-Einstellungen
- Operationen pro CAN-Linie
- Operation für das Objektverzeichnis
- Operationen für die Kommunikationsparameter
- Operationen für die herstellerspezifischen Parameter
- Operationen für die Geräteprofilparameter

Hardwarekonfigurationen können:

- hinzugefügt,
- kopiert,
- gelöscht
- importiert und exportiert werden.

Für die einzelne CAN-Linie lassen sich ausführen:

- Hinzufügen einer weiteren Linie
- Löschen der gesamten CAN-Linie

Für das Objektverzeichnis werden folgende Aktionen angeboten:

- Datenimport aus einer Profildatei
- Datenimport aus einer EDS/XDD-Datei
- Datenexport in eine Profildatei

Für das Kommunikationsparametersegment stehen die folgenden Aktionen zur Verfügung:

- Datenimport aus einer Datei (Profil oder EDS/XDD)
- Datenexport in eine Profildatei
- Anlegen neuer Objekte (Indizes)
- Kommunikationsdienstauswahl zum Anlegen dienstbezogener Objekte

Das herstellerspezifische Segment hat Aktionstasten für:

- Datenimport aus einer Datei (Profil oder EDS/XDD)
- Datenexport in eine Profildatei
- Anlegen neuer Objekte (Indizes)
- Anlegen eines Subsegments

Für das Geräteprofilsegment sind folgende Aktionen möglich:

- Datenimport aus einer Datei (Profil oder EDS/XDD)
- Datenexport in eine Profildatei
- Anlegen neuer Objekte (Indizes)
- Anlegen eines Subsegments

Subsegmente lassen folgende Aktionen zu:

- Datenimport aus einer Datei (Profil oder EDS/XDD)
- Datenexport in eine Profildatei
- Anlegen neuer Objekte (Indizes)
- Konfiguration des Subsegments

2.4. Eingabeformulare

Eingabeformulare bzw. Eingabemasken vereinfachen die Eingabe bestimmter Daten. Neben den nachfolgenden Beschreibungen ist für jede Maske eine Hilfe mit weiterführenden Beschreibungen über <F1> aufrufbar.

2.4.1. Allgemeine Einstellungen

Die allgemeinen Einstellungen (*General Settings*) dienen der Definition globaler Geräteparameter. Anhand dieser Parameter wird die CANopen-Bibliothek über die Datei **cal_conf.h** konfiguriert.

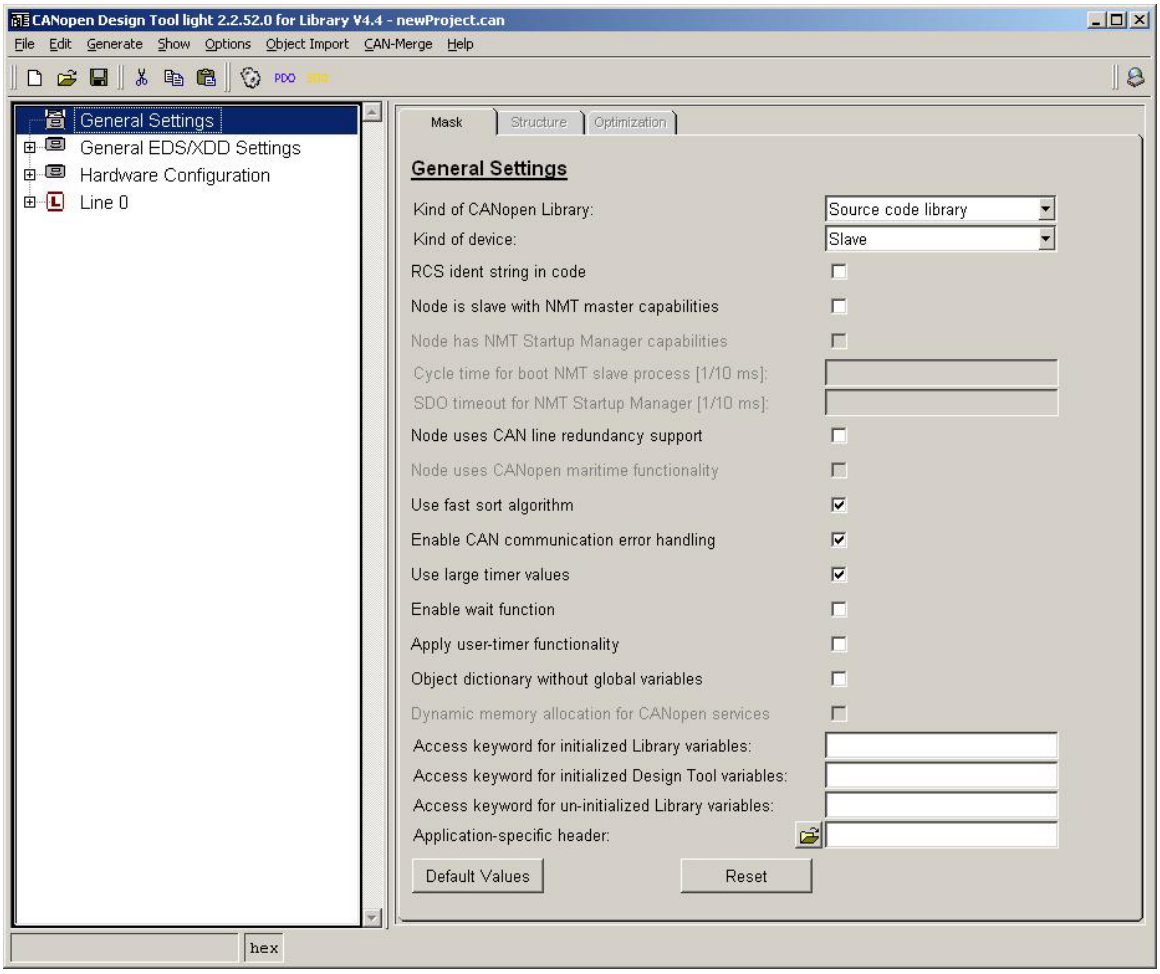

CANopen Design Tool light 2.2.52.0 for Library V4.4 - newProject.can
File Edit Generate Show Options Object Import CAN-Merge Help
General Settings
General EDS/XDD Settings
Hardware Configuration
Line 0
Mask
Structure
Optimization
General Settings
Kind of CANopen Library:
Source code library
Kind of device:
Slave
RCS ident string in code
Node is slave with NMT master capabilities
Node has NMT Startup Manager capabilities
Cycle time for boot NMT slave process [1/10 ms]:
SDO timeout for NMT Startup Manager [1/10 ms]:
Node uses CAN line redundancy support
Node uses CANopen maritime functionality
Use fast sort algorithm
Enable CAN communication error handling
Use large timer values
Enable wait function
Apply user-timer functionality
Object dictionary without global variables
Dynamic memory allocation for CANopen services
Access keyword for initialized Library variables:
Access keyword for initialized Design Tool variables:
Access keyword for un-initialized Library variables:
Application-specific header:
Default Values
Reset
hex

Abbildung 3, Maske für die allgemeinen Einstellungen

Die Defaultwerte sind optimal für einen CANopen-Slave.
Die einzelnen Punkte sind in der Kontexthilfe erläutert.

2.4.2. Allgemeine EDS/ESI/XDD-Einstellungen

Die globale EDS/ESI/XDD-Maske wird zum Bearbeiten aller geräteglobalen EDS/ESI/XDD-Daten verwendet. Das sind im wesentlichen Geräte-, Hersteller- und Bestellbezeichnungen.

2.4.3. Advanced configuration

In dieser Maske können anwenderspezifische Compiler-Directiven für die Konfigurationsdatei **cal_conf.h** eingetragen werden.
Bitte geben Sie gültigen C-Code ein. Ein Beispiel:

```
#define CONFIG_BIG_RAM_VERSION 1
/* undefine printf */
```

```
#define PRINTF
```

2.4.4. Hardwarekonfigurationen

Eine Hardwarekonfiguration kann über ihr Aktionsmenü umbenannt, dupliziert, gelöscht, importiert und exportiert werden. Die aktive Hardwarekonfiguration wird im Projektbaum durch ein Paar von Sternchen (*) gekennzeichnet. Die Unterpunkte *Debug Settings*, *CPU Settings* und *Compiler Settings* sind einmal pro Hardwarekonfiguration vorhanden. Dagegen muss der Abschnitt *CAN Settings* für jede einzelne CAN-Linie im Projekt vorhanden sein.

Die Bedeutung der Parameter der einzelnen Unterpunkte wird in der Online-Hilfe, welche über F1 oder das Menü "Help -> Context Help" aufgerufen werden kann, erläutert.

2.4.5. Linienspezifische EDS/XDD-Einstellungen

Mit der EDS/XDD-Maske für die einzelnen CAN-Linien können Eigenschaften des Gerätes beschrieben werden, wovon einige nur in die XDD-Datei übernommen werden. Weiterhin kann ein Kommentar für die EDS-Datei eingegeben werden.

2.4.6. Linienspezifische Standardeinstellungen

Hier kann bestimmt werden, ob die Linie als Slave oder als Master fungieren soll. Weiterhin lassen sich hier globale Vorgaben für die Optimierungen der einzelnen Objekte ab 2000_h festlegen. Der Standardfall, dass eine Variable für ein Objekt angelegt werden soll, und dass die Beschreibungsstruktur, jedoch nicht die Variable, kontant ist, ist voreingestellt. Detailierte Informationen zu den einzelnen Punkten findet sich in der Kontexthilfe der Optimierungsregister der Indizes. Bitte beachten Sie, dass Änderungen für das gesamte Objektverzeichnis nur vorgenommen werden, wenn sich Daten in der Maske ändern. Z.B.: 'Create Variable' ist für alle Objekte gesetzt, und bei einem Objekt wird es von Hand deaktiviert. Soll nun 'Create Variable' wieder für alle gesetzt werden, so ist die 'Standard Settings'-Maske auszuwählen, 'Create Variable' zu deaktivieren, die Maske erneut auszuwählen und 'Create Variable' erneut zu setzen. Erst jetzt ist wieder bei allen Objekten das Attribute 'Create Variable' gesetzt.

2.4.7. Zusätzliche CANopen Einstellungen

Die zusätzlichen CANopen Einstellungen dienen der Konfiguration von Zusatzdiensten der Bibliothek, welche über den normalen Lieferumfang hinaus gehen. Bitte beachten Sie das Handbuch der Bibliothek und die Kontexthilfe für weitere Informationen.

2.4.8. Maskenansicht

Die Maskenansicht ist eine Eingabehilfe für die Kommunikationsparameter. Damit wird die Eingabe der Parameter erleichtert, da die teilweise bitkodierten Werte in Klartextausgaben umgewandelt werden. Zusammenhängende Objekte werden ebenfalls transparent für den Bediener parametriert. Bitte beachten Sie, dass die Maskenansicht nur für einige Objekte zur Verfügung steht.

2.4.9. Strukturansicht

Die Strukturansicht ist der allgemeingültige Editor, mit dem alle Objektparameter verändert werden können. Es wird unterschieden in eine Ansicht für die Index-Parameter und eine für die Sub-Index-Parameter.

Bezeichnung	**Beschreibung**
Index	Nummer des Index' (hexadezimal)
Name	Name der globalen C-Variable
Datentyp	Datentyp entsprechend dem CANopen-Standard
Object Code	Art des Objekts nach dem CANopen-Standard: Mögliche Werte: Variable, Array, Record, Domain
Object-Name	Name des Objekts in der XDD/EDS/ESI-Datei
Description	Beschreibung des Objekts für die Dokumentation und die Parameterbeschreibung in XDD-Dateien

Tabelle 1, Elemente der Strukturansicht für Indizes

Für Objekte im Kommunikationssegment (1000_h - $1FFF_h$) sind die Datentypen und ObjectCodes nicht veränderbar, da diese Werte durch den Standard vorgegeben sind.

Bezeichnung	Beschreibung	Lib	EDS	XDD	Doc
Index	Nummer des Index (hexadezimal)	x	x	x	x
Sub-Index	Nummer des Sub-Index (dezimal)	x	x	x	x
PDO-Mapping	wenn selektiert, kann das aktuelle Objekt einem PDO zugeordnet werden	x	x	x	x
EDS-Name	EDS-Name für Sub-Index (nur relevant für Arrays und Records)		x	x	x
Unit	Einheit des Objektes				x
Access	Zugriffsrecht auf das Objekt vom CAN aus	x	x	x	x
Data Type	Typ des Objektes	x	x	x	x
Size	Größe des Objektes in Byte	x			
Value	Vorbelegungswert der Variable	x	x	x	x
Lower Limit	unteres Limit für nummerische Werte	x	x	x	x
Upper Limit	oberes Limit für numerische Werte	x	x	x	x
DefaultValue in EDS*	Vorbelegungswert im EDS speichern		x	x	
Limits in EDS*	Limits im EDS speichern		x	x	
Refuse read on scan*	Gleichnamige ObjFlags-Einstellung in EDS/XDD-Dateien		x	x	
Refuse write on download*	Gleichnamige ObjFlags-Einstellung in EDS/XDD-Dateien		x	x	
Valid after reset*	Gleichnamige ObjFlags-Einstellung in XDD-Dateien			x	

Tabelle 2, Elemente der Strukturansicht für Sub-Indizes

Alle mit einem Stern (*) markierte Elemente werden selten benötigt und können daher über "Options → View Options → Hide EDS/XDD flags" ausgeblendet werden.

Innerhalb der *Object Name* und *Name* Felder können interne %-Variablen verwendet werden. Diese werden bei der Generierung der Programmausgaben durch ihren aktuellen Wert ersetzt. %s beinhaltet z.B. den aktuellen Sub-Index und ein EDS-Name wie "Ausgang %s" wird je nach Sub-Index in "Ausgang 0", "Ausgang 1" ... oder "Ausgang 254" ersetzt. Damit vereinfacht sich das Kopieren und Duplizieren von Objekten und nummerische Werte in Variablen- und EDS-Namen müssen nicht nachträglich mühsam von Hand bearbeitet werden. Eine Zusammenstellung aller %-Variablen und Hinweise zu deren Verwendung befindet sich im Abschnitt *%-Variablen* in Kapitel 4.7.

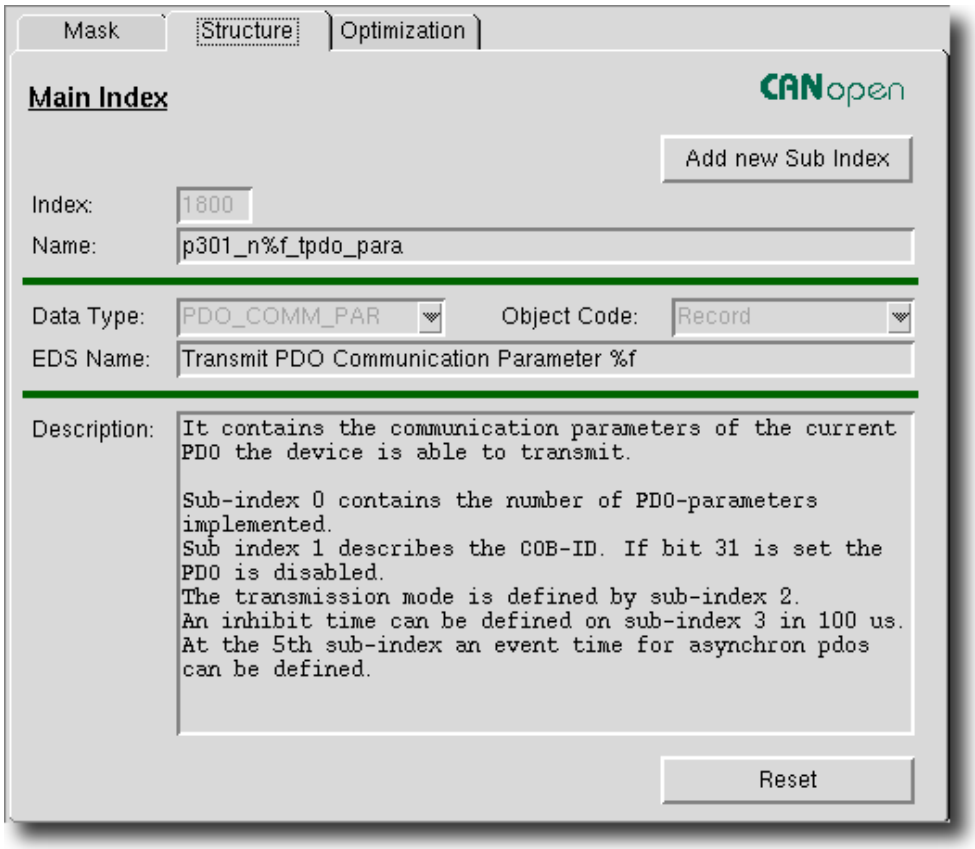

Mask
Structure
Optimization
Main Index
CANopen
Add new Sub Index
Index:
1800
Name:
p301_n%f_tpdo_para
Data Type:
PDO_COMM_PAR
Object Code:
Record
EDS Name:
Transmit PDO Communication Parameter %f
Description:
It contains the communication parameters of the current PDO the device is able to transmit.
Sub-index 0 contains the number of PDO-parameters implemented.
Sub index 1 describes the COB-ID. If bit 31 is set the PDO is disabled.
The transmission mode is defined by sub-index 2.
An inhibit time can be defined on sub-index 3 in 100 us.
At the 5th sub-index an event time for asynchron pdos can be defined.
Reset

Abbildung 4, Strukturansicht eines Objekts

2.4.10. Optimierungsansicht

Die Optimierungsansicht bietet Zugriff auf Optimierungsparameter der Objekte. Die Optimierungen wirken sich ausschließlich auf die generierten C-Dateien und damit auf den Speicherbedarf der Library aus.

Storage Class
: dient der Angabe einer Speicherklasse für das Objekt. Diese kann z.B.: IDATA, XDATA, CO_CONST oder ein anderer compiler-abhängiger Wert sein.

Create Variable
: Ist dieser Parameter gesetzt, so wird eine Variable mit dem angegebenen Variablennamen angelegt. Anderenfalls wird die Variable in der Datei **objects.h** als extern deklariert, falls *create extern declaration* aktiv ist.

Same Structure for all Subs
: Ist dieser Wert gesetzt, wird der 'Short Description'-Parameter in der Beschreibungsstruktur gesetzt. (nur relevant für Arrays)

Constant Value Description Structure
: Ist dies aktiviert, so wird die Beschreibungsstruktur als konstant deklariert. Ändert sich diese während der Laufzeit, so darf die Struktur nicht konstant sein.

Virtual Object
Ist ein Objekt als virtuelles Objekt gekennzeichnet, so erscheint es nicht im Objektverzeichnis in der Datei **objects.h**, jedoch in der Dokumentation und in der EDS-Datei. Dies ist per Default deaktiviert. Dieses Attribut steht nur für Objekte des herstellerspezifischen Bereichs oder des Geräteprofilbereichs zur Verfügung.

Create Type Definition
Dieses Attribute ist nur bei benutzerdefinierten Datentypen zugänglich und aktiviert die Typdefinition für diesen Datentyp in der Datei `objects.h`.

Constant default value
Ist dieses Attribute gesetzt, so werden die Vorbelegungswerte des Objekts in einem konstanten Array gespeichert.

Constant limits
Ist dieses Attribute gesetzt, so werden die Grenzwerte des Objekts in einem konstanten Array gespeichert.

In den meisten Fällen sind die Defaultwerte und Grenzwert der Objekte konstant. Für spezielle Anwendungsfälle (verschiedene Konfigurationen je nach DIP-Switch, ..) können diese Werte variabel sein. Dabei wird jedoch mehr Speicher benötigt. Bei Strings und Domains sind diese Einstellungen ohne Effekt.

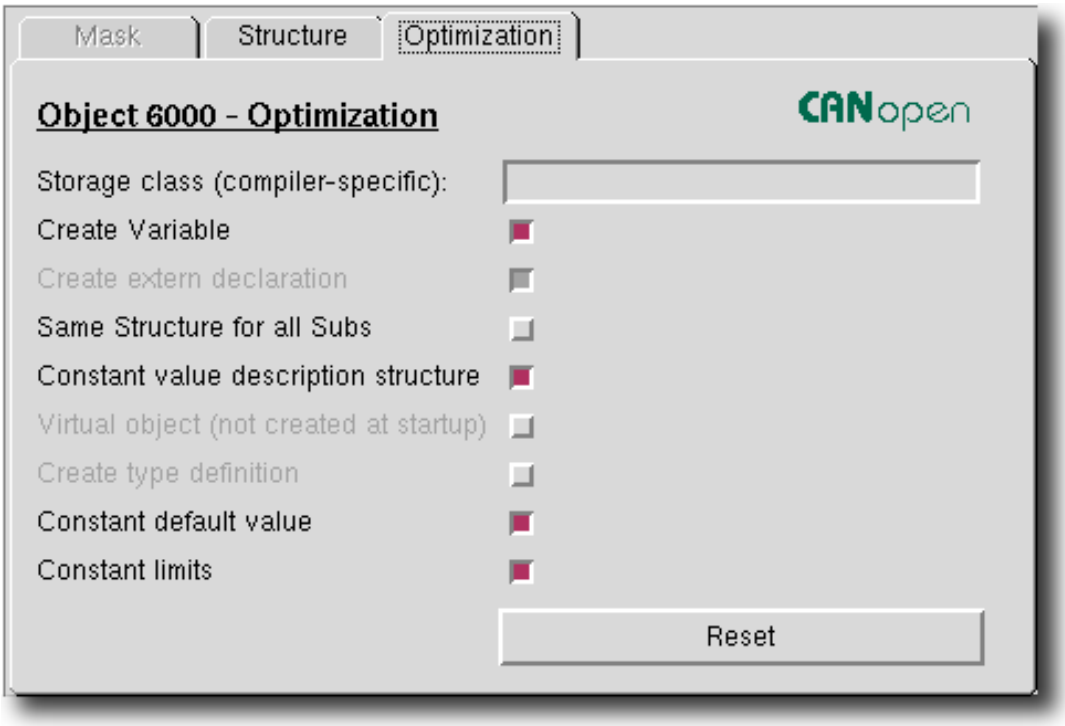


Abbildung 5, Optimierungsansicht eines Objekts

2.5. Programmsteuerung

Die Bedienung des gesamten Programmes erfolgt weitgehend über die Maus. Die in die Eingabefelder eingetragenen Werte werden erst beim Verlassen der Maske übernommen,

d.h. zusammenhängende Aktionen wie z.B. Berechnung neuer Größen bei Typwechsel werden dann erst ausgeführt.

Änderungen an einer Maske können, bevor die Maske verlassen wird, über den Button "Reset" zurückgesetzt werden. Zudem ermöglicht der Button "Default" das Zurücksetzen auf Vorbelegungswerte, falls für diese Maske Vorbelegungswerte verfügbar sind.

3. Dateistruktur

3.1. Projektstruktur

3.1.1. Projektdatei

Alle Daten eines Projektes werden in einer Projektdatei zusammengefasst. Dennoch ist es sinnvoll für jedes Projekt ein separates Verzeichnis zu verwenden, da die generierten Dateien in dem jeweiligen Verzeichnis der Projektdatei erstellt werden.

Im Verzeichnis 'projects' unterhalb des Programmverzeichnisses befinden sich einige Beispielprojekte.

3.1.2. Generierte Dateien

Die generierten Dateien sind in Tabelle 3 zusammengefasst.

Datei	Beschreibung
objects.h	Objektverzeichnisimplementierung (C-Code)
objects.c	Objektverzeichnisimplementierung (C-Code) (optional)
cal_conf.h	Bibliothekskonfigurationsdatei (C-Code)
co_init.c	Initialisierungsdatei für die Bibliothek (C-Code)
<EDSDateiName>.eds	erzeugte EDS-Datei für CAN-Linie
<EDSDateiName>.txt	Beschreibungsdatei für den CANopen Device Monitor
<EDSDateiName>.xdd	erzeugte XDD-Datei für CAN-Linie im XML-Format
<Projektname>_docu.txt	Text-Dokumentation der implementierten Objekte
<Projektname>.html	Html-Dokumentation der implementierten Objekte
userList.txt	anwenderspezifische Objektbeschreibung (optional)
generate.err	Informationsdatei, enthält Fehlermeldungen und Warnungen, welche während der Codegenerierung auftraten

Tabelle 3, Generierte Dateien

Der CANopen EDS Editor erzeugt die XDD- und EDS-Datei sowie die Dokumentation, jedoch keinen C-Code.
Nach der Generierung stehen die EDS/XML-Dateien, das Objektverzeichnis, die Initialisierungs- und die Konfigurationsdatei für die CANopen-Bibliothek und eine Dokumentation zur Verfügung. Falls die Generierung fehlschlägt, so werden die Dateien nicht erzeugt und in der Datei **generate.err** befinden sich die Fehlermeldungen.

3.2. Profildateien

Profildateien beinhalten Teile eines Objektverzeichnisses mit den jeweiligen Objekten und ihren Attributen. Das CANopen Design Tool wird standardmäßig mit einem Profil, das alle Kommunikationsobjekte enthält, ausgeliefert. Werden Kommunikationsobjekte mit dem CANopen Design Tool angelegt, so werden die Einträge automatisch aus diesem Profil geladen. Weitere Profildateien für die CANopen-Geräteprofile sind bei *port* verfügbar. Mittels dieser Profile lassen sich standardgerechte Applikation in kürzester Zeit realisieren.

Zu Demonstrationszwecken sind einige Geräteprofildatenbanken bereits dem CANopen Design Tool beigelegt. Objekte aus diesen Geräteprofildatenbanken können zwar importiert werden, jedoch ist die Generierung der Ausgaben nur mit der entsprechenden Lizenz für die Geräteprofildatenbank möglich. Bei Fragen zu Ihrer Lizenz kontaktieren Sie bitte service@port.de ⟨service@port.de⟩.

Sollen Teile eines Projektes in späteren Projekten wiederverwendet werden, so können diese Daten in eine separate Profildatei exportiert werden.

3.2.1. Profil CiA-443: SIIS level-2 devices

Dateiname der Profil-Datenbank:	profile443_v2_1_0.pro
Verfügbarkeit:	Lizenz notwendig
CiA Standard-Referenz:	CiA-443, V2.1.0

Die Profil-Datenbank zum CiA-443 unterstützt die profil-spezifischen Datentypen und Objekte mit Ausnahme von Objekt 6010h (p443_cpu_clock_time_and_date).

Die profil-spezifischen Datentypen sind von der Anzahl der Kanäle bzw. Ventile abhängig und können von Objekt zu Objekt verschieden sein. Das Design Tool generiert spezielle C Strukturen für die kanal-abhängigen Subindizes, um einen effektiven Zugriff auf die Objekte im C Code zu ermöglichen. Alle Struktur-Definitionen sind in der generierten Datei objects.h gespeichert.

```
Beispiel: Datentyp 0082h (12-CHANNEL METAL LOSS)

Der Datentyp setzt sich aus allgemeinen und kanal-abhängigen Subindizes zusammen.
Es wird folgende C Struktur für die kanal-abhängigen Subindizes generiert:

typedef struct {
    UNSIGNED8  status;                  /**< channel status */
    REAL32     averageMetalLoss;        /**< channel average metal loss */
    REAL32     averageTemperature;      /**< channel average temperature */
    REAL32     sector_1_metalLoss;      /**< channel sector 1 metal loss */
    REAL32     sector_1_temperature;    /**< channel sector 1 temperature */
    REAL32     sector_2_metalLoss;      /**< channel sector 2 metal loss */
    REAL32     sector_2_temperature;    /**< channel sector 2 temperature */
    REAL32     sector_3_metalLoss;      /**< channel sector 3 metal loss */
    REAL32     sector_3_temperature;    /**< channel sector 3 temperature */
```

```
    REAL32      sector_4_metalLoss;      /**< channel sector 4 metal loss */
    REAL32      sector_4_temperature;    /**< channel sector 4 temperature */
    REAL32      sector_5_metalLoss;      /**< channel sector 5 metal loss */
    REAL32      sector_5_temperature;    /**< channel sector 5 temperature */
    REAL32      sector_6_metalLoss;      /**< channel sector 6 metal loss */
    REAL32      sector_6_temperature;    /**< channel sector 6 temperature */
    REAL32      sector_7_metalLoss;      /**< channel sector 7 metal loss */
    REAL32      sector_7_temperature;    /**< channel sector 7 temperature */
    REAL32      sector_8_metalLoss;      /**< channel sector 8 metal loss */
    REAL32      sector_8_temperature;    /**< channel sector 8 temperature */
} P443_CHANNEL_METAL_LOSS_T;
```

Die C Struktur für den Datentyp 0082h mit 2 Kanälen ist wie folgt definiert:

```
typedef struct {
    UNSIGNED8       numOfEntries;       /**< number of entries */
    UNSIGNED32      siUnitMetalLoss;    /**< SI unit for metal loss */
    UNSIGNED32      siUnitTemperature;  /**< SI unit for temperature */
    P443_CHANNEL_METAL_LOSS_T chan[2];  /**< channels */
} P443_12_CHANNEL_METAL_LOSS_2_T;
```

Voraussetzung für die Generierung der kanal-abhängigen C Strukturen ist die Angabe der Profilnummer in Objekt 1000h (p301_device_type) oder bei Implementierung mehrerer logischer Geräte in Objekt 67FFh (single_device_type).

Die Namen der Elemente in den C Strukturen können nicht geändert werden.

Die C Namen der P443 Objekte sind im Tab Structure des Main Index zu ändern, wenn diese Objekte mehrmals importiert werden und keine Segmentierung verwendet wird.

4. Projektbearbeitung

Dieses Kapitel beschreibt die Bearbeitungsfolge zur Erstellung und Bearbeitung von Projekten. Die vorgeschlagene Reihenfolge ist nicht bindend, aber zweckmäßig für die Beschreibung. Zu Beginn der Entwicklung müssen grundlegende Fragen, wie die Anzahl der CAN-Linien und die Art des Gerätes (Slave oder Master) bereits geklärt sein. Die notwendigen Arbeitsschritte sind die folgenden:

- globale Einstellungen konfigurieren
- globale EDS bzw. ESI-Parameter konfigurieren
- die Hardwareeinstellungen konfigurieren
- EDS-Parameter für CAN-Linie konfigurieren
- Standard- und zusätzliche Einstellungen konfigurieren
- Applikationsvariablen definieren
- Applikationsvariablen parametrieren
- Kommunikationsvariablen definieren
- Kommunikationsvariablen parametrieren
- ggf. Optimierung der einzelnen Objekte
- Generierung der Ausgaben

4.1. Projekterzeugung

Ein neues Projekt wird über das Menü *File* → *New Project* angelegt.
Existierende Projekte werden über die Kurztaste *Open Project* oder über das Menü *File* geöffnet.
Jede CAN-Linie kann nun mit Daten aus einem Profil oder einer EDS-Datei vorbelegt werden.

4.2. Hardwarekonfiguration

Es ist empfohlen, dass zunächst die Hardware, auf welcher später die Applikation ausgeführt werden soll, konfiguriert wird. Die wichtigste Entscheidung ist die Wahl der CPU bzw. des Betriebssystems.
Defaultkonfigurationen für die unterstützten Plattformen befinden sich beim s1-Beispiel der CANopen-Bibliothek. Diese Konfigurationsdateien (`conf_xxxx.h`) können über "Import Configuration" in ein Projekt importiert werden.

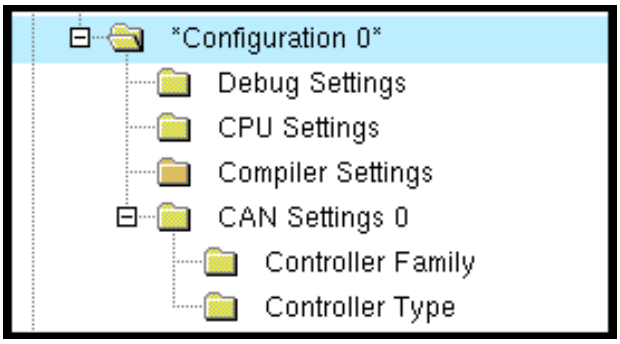


Abbildung 6, Hardwarekonfiguration im Projektbaum

Ist dies unter *CPU Settings* eingestellt, so können die zusätzlichen Parameter der CPU in der Mehrzahl aller Anwendungen auf ihren Defaultwerten bleiben. Bei *Compiler Settings* kann nun der verwendete Compiler eingestellt werden. Auch hier sind die Defaultwerte für die meisten Anwendungsfälle gültig. Bei den *CAN Settings* muss zunächst die CAN-Familie und daraus ein CAN-Controller-Typ ausgewählt werden. Hat das Projekt mehr als eine CAN-Linie, so müssen die CAN-Einstellungen für jede Linie angelegt werden. Dabei entsprechen die Nummer hinter den *CAN Settings* der jeweiligen Linie. Detailierte Informationen zu jedem Parameter bietet die Online-Hilfe.

Soll die Anwendung auf mehreren Hardwareplattformen eingesetzt werden, so können mehrere Hardwarekonfigurationen angelegt werden. Mit *Mark configuration as active* kann jeweils die aktive Konfiguration festgelegt werden. Nur die aktive Konfiguration wird bei der Generierung der Ausgaben in der Datei **cal_conf.h** aktiviert. Ist keine Konfiguration aktiviert, so muss das entsprechende Define `CONFIG_USE_TARGET_x` im Makefile oder im Compiler auf 1 gesetzt werden. Die Nutzung mehrerer Konfigurationen ist auch sinnvoll, wenn man eine funktionierende Konfiguration behalten will und mit einer anderen experimentieren möchte.

4.3. Applikationsvariablen

Als Applikationsvariablendefinition wird das Anlegen von Objekten mit Indizes von $2000_h - 5FFF_h$ für herstellerspezifische Profile oder das Anlegen von Objekten mit den Indizes von $6000_h - 9FFF_h$ der standardisierten Geräteprofile bezeichnet. Es gibt zwei Wege die Applikationsvariablen zu definieren:

- Laden aus einem Profil
- Anlegen von Hand

Der erste Weg ist der einfachste, in dem einzelne Objekte mittels ihres Index' aus einem Profil geladen werden. Falls noch keine Profile existieren, lassen sich alle Einträge per Hand erstellen. Bei der Erstellung wird unterschieden in Parameter, die den Index beschreiben und in Parameter, die die Sub-Indizes beschreiben. Die Indexparameter definieren das Layout der Variable. Das heißt, den Typ, die Art der Variable (Variable, Array, Record, Domain), die Anzahl der Sub-Indizes , den C-Variablennamen und den Objekt-Namen. Weiterhin kann jede Variable mit einem Text beschrieben werden. Diese

Beschreibung wird in der Dokumentation eingefügt. Bei den Parametern der Sub-Indizes kann ein Objekt-Name zur symbolischen Beschreibung der Record- bzw. Arrayelemente vergeben werden. Bei Records kann der Typ der Recordelemente und ihre Größe definiert werden. Weiterhin ist es möglich, eine Einheit für die Werte anzugeben. Bei der Verwendung eigener Datentypen für Records sind diese vorher im 'Data Types'-Abschnitt der Linie zu definieren.

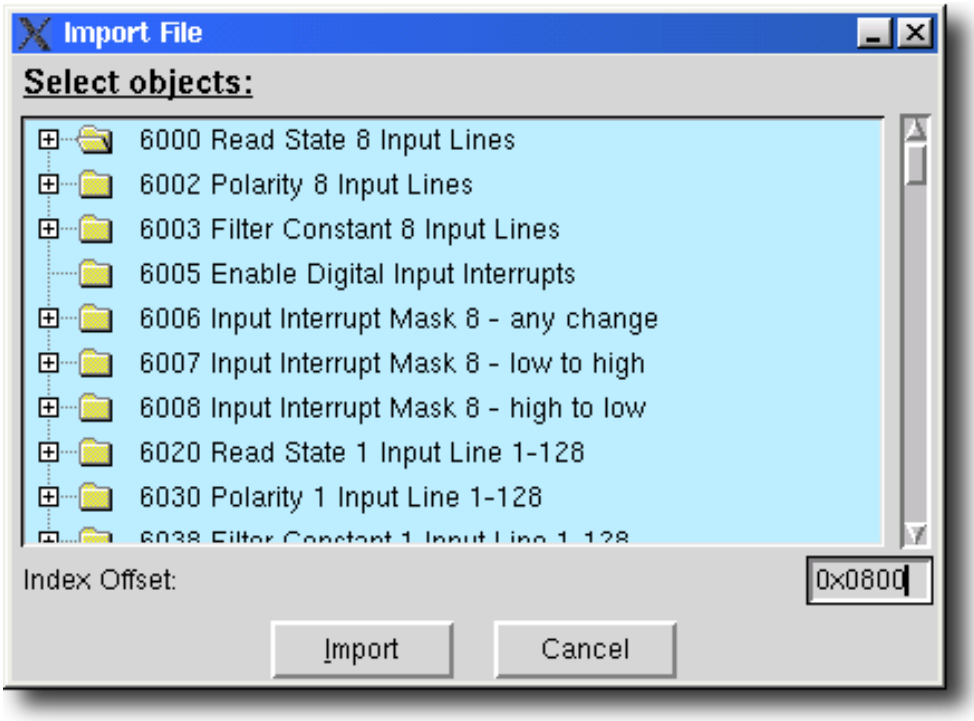

Import File
Select objects:
6000 Read State 8 Input Lines
6002 Polarity 8 Input Lines
6003 Filter Constant 8 Input Lines
6005 Enable Digital Input Interrupts
6006 Input Interrupt Mask 8 - any change
6007 Input Interrupt Mask 8 - low to high
6008 Input Interrupt Mask 8 - high to low
6020 Read State 1 Input Line 1-128
6030 Polarity 1 Input Line 1-128
Index Offset:
0x0800
Import
Cancel

Abbildung 7, Importmaske für Profile

4.3.1. Parametrierung von Applikationsvariablen

Die Applikationsvariablenparametrierung findet in der Strukturansicht für die Sub-Indexelemente statt. Hier werden den Variablen Standardwerte zugewiesen. Diese Parametrierung kann gleichzeitig mit der Objektdefinition stattfinden.

4.4. Kommunikationsvariablen

Bei der Kommunikationsvariablendefinition sind die gleichen Mechanismen wie bei den Applikationsvariablen möglich. Im Unterschied dazu werden die notwendigen Daten aus der Kommunikationsobjektdatenbank geladen. Das bedeutet, daß die Struktur bereits definiert ist und Namen vorbelegt sind. Für ein CANopen-kompatibles Gerät sind mindestens die Objekte

- Device Type (1000_h)
- Identity Object (1018_h)

und weitere

zu implementieren. Diese Einträge werden von CANopen Standard gefordert.

Für das Anlegen von Kommunikationsparametern für die CANopen-Dienste stehen spezielle Menüs zur Verfügung. Diese sind über die Taste *Add new Communication Service* im Aktionsmenü des *Communication Segmegnts* erreichbar.

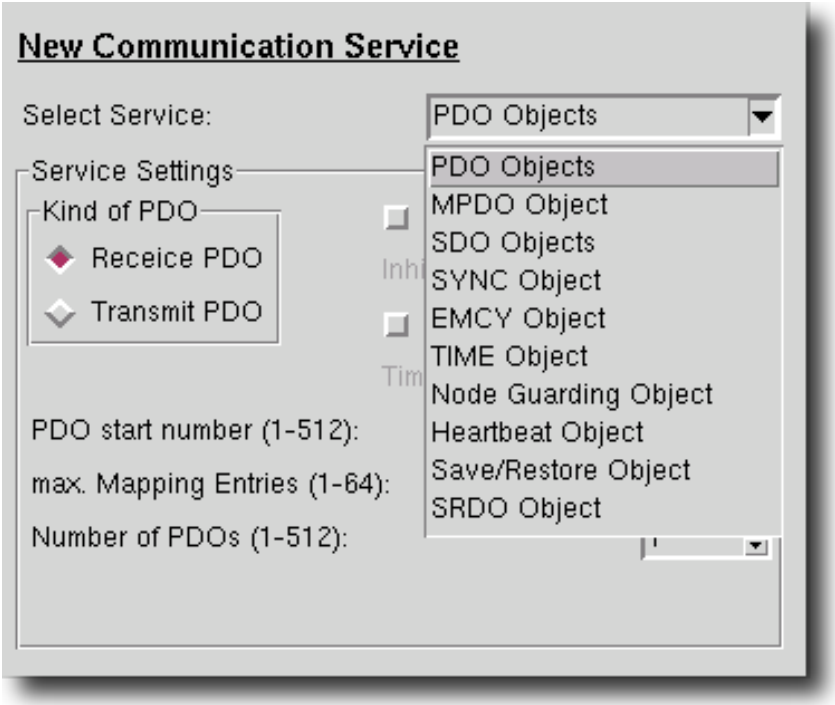


Abbildung 8, Dialog zum Anlegen neuer Kommunikationsdienste

Für die Parametrierung der CANopen Dienste, z.B. PDOs sind spezielle Menüs vorgesehen. Der Anwender muß in diesen Menüs nur Designentscheidungen fällen und benötigt keinerlei Kenntnisse über die Kodierung und Objektzusammenhänge für die Kommunikation.

4.4.1. Parametrierung

Für die Parametrierung der Kommunikationsobjekte kann die Maskenansicht genutzt werden. Mit diesen Masken ist es sehr einfach, komplexe Kommunikationsparameter wie PDO-Parameter und PDO-Mapping zu parametrieren.

4.5. Generierung des Quellcodes

Beim Betätigen der Taste *Generate all Files* im Menu *Generate* werden eine Objektverzeichnisimplementierung in C-Code **(objects.h, objects.c)**, die Initialisierungs- und Konfigurationsdateien sowie eine EDS-Datei für jede CAN-Linie **(**<*ProjektName*>*line*<*LinienNr*>**.eds** und, falls aktiviert, eine Dokumentation **(**<*projektname*>**.html)** erzeugt. Diese Dateien sind immer konsistent und beinhalten das Datum und die Uhrzeit der Generierung. Die Datei **generate.err** enthält Warnungen und Fehlermeldungen, welche

bei der Generierung auftraten. Es wird empfohlen, sich nach jeder Generierung den Inhalt dieser Datei mit *Show* anzeigen zu lassen.

4.6. %-Variablen

Die %-Variablen können in Objekt-Namen und Variablennamen der Indizes und in Objekt-Namen der Subindizes benutzt werden. In der Datei **objects.c**, in den EDS-Dateien und der Dokumentation und auch im Objektbaum werden sie durch ihre aktuellen Werte ersetzt.

Variable	Inhalt
%k	number of sub segment, starting at 1
%i	Nummer des Index
%s	Nummer des Subindex
%l	Nummer der Linie
%f	Nummer des Dienstes (SSDO, CSDO, RPDO, RPDO-Mapping, TPDO-Mapping, SRDO, SRDO-Mapping) beginnend mit 1
%p	(Index - Index des Segmentsanfangs) beginnend mit 0
%q	(Index - Index des Segmentsanfangs +1) beginnend mit 1
%u	(Index - Index des Subsegmentsanfangs) beginnend mit 0
%v	(Index - Index des Subsgmentsanfangs +1) beginnend mit 1
%k	Nummer des Subsegments beginnend mit 1
%t	Kurzbezeichnung des Datentypen

Tabelle 5, %-Variablen

Die Variablen als Kleinbuchstaben geben jeweils einen dezimalen Wert zurück. Wird die Variable mit großen Buchstaben geschrieben, so wird ein hexadezimaler Wert ohne führendes "0x" zurückgegeben. Variablen, welche im aktuellen Kontext undefiniert sind, (z.B. %s bei einem Index) geben einen leeren String zurück.

Beispiele für eine Anwendung der %-Variablen findet man u.a. bei den SDO und PDO - Einträgen im mitgelieferten Kommunikationsprofil.

4.7. {}-Ausdrücke

Die {}-Ausdrücke ermöglichen die Verwendung von %-Variablen und mathematischen Operatoren. Innerhalb der geschweiften Klammern können die %-Variablen, Konstanten (dezimal/hexadezimal) und die Operatoren + - * und / verwendet werden. Der gesamte Ausdruck wird bei der Generierung durch den resultierenden numerischen Wert ersetzt. Dieser Rückgabewert ist immer dezimal.
Beispiele für gültige Objekt-Namen mit {}-Ausdrücken:

- output {%i - 0x100}
- state {%p + 100} at device %l

4.8. Programmaufrufe nach der Generierung

Via *Generation* → *Post-Generation Settings* lässt sich ein Kommando/Programm konfigurieren, welches nach Abschluss der Generierung aufgerufen wird. Das Kommando kann ein Programm, ein Shell-Skript, eine Batch-Datei oder ein Aufruf wie z.B. 'make all' sein. Als Arbeitsverzeichnis wird das Arbeitsverzeichnis des CANopen Design Tool genutzt.

5. Sonderfälle

Dieses Kapitel beschreibt einige Sonderfälle zur Objektverzeichnisgenerierung. Das CANopen Design Tool unterstützt die Generierung von Objektverzeichnisimplementierungen für die Ein-Linien-Version und die Mehrlinien-Version der CANopen Library von *port*. Die nachfolgenden Sonderfälle dienen zur Optimierung der Implementierung.

5.1. Bereichsüberwachung von Variablenwerten

Bei der Generierung des Objektverzeichnisses werden standardmäßig Einträge für Minimal- und Maximalwerte angelegt. Diese Grenzwerte werden bei der CANopen Library von *port* beim SDO-Transfer zur Grenzwertüberwachung genutzt. Die Überwachung der Grenzwerte kann bei der Übersetzung der Applikation über Objekt 1200h / Tab Mask / Checkbox "Check Object Limits with SDO Transfer" an- oder abgeschaltet werden. Diese Einstellung aktiviert oder deaktiviert das Compiler-Define **CONFIG_LIMITS_CHECK**.

5.2. Gruppierung von Indizes in Subsegmenten

Eine weitere Eigenschaft ist die Organisation der Gerätevariablen innerhalb von Strukturen. Eine solche Organisation ist möglich für die herstellerspezifischen Daten und die Daten der standardisierten Geräteprofile. Zweckmäßig ist eine solche Organisation, wenn die Einträge der einzelnen Segmente identisch sind. Sollen die Daten zweier Segmente in einer Struktur abgelegt werden, so müssen die Daten in zwei Subsegmenten im identischem Variablennamen und identischer Länge abgelegt werden.

Zweckmässiges Vorgehen ist hierbei:

- Anlegen des Subsegments (Add new Sub Segment)
- Füllen des Subsegments mit Objekten
- Duplizieren des Subsegments (Duplicate)

Das Ergebnis dieser Gruppierung ist eine Struktur und ein C-Array vom Typ dieser Struktur. Das Array hat so viele Elemente wie Gruppenmitglieder (Segmente mit gleichem Variablennamen) vorhanden sind. Der Vorteil einer solchen Organisation ist, daß die indizierte Adressierung von C-Feldern genutzt werden kann. Vorstellbar ist dies z.B. beim Einsatz in Mehrachsantrieben zum schnellen Wechsel zwischen den einzelnen Achsen in der Firmware.

Bei den herstellerspezifischen Daten wird das gleiche Prinzip verfolgt. Dort ist es allerdings möglich, einen Startindex für die Segmentierung und die Segmentgröße frei zu wählen. Das bedeutet, daß bis zu einem bestimmten Index die Daten ganz normal abgelegt werden. Ab diesem Index wird der Baum im herstellerspezifischen Bereich segmentiert.

5.3. Herstellerspezifische Strukturen (Datentypen)

Um im herstellerspezifischen Bereich Records mit eigenen Strukturen (Datentypen) zu nutzen, müssen diese Datentypen zunächst im 'Data Types' Abschnitt der Linie angelegt werden. Dazu wird mit 'Add new Data Type' ein freier Index zwischen 0040_h und

$005F_h$ mit dem Objektcode 'DefStruct' angelegt. Der Namen der Struktur wird bei 'Name' und der EDS-Name bei 'EDS-Name' eingetragen. Der Name, z.b. 'MYTYPE_T' wird in der Datei **objects.h** verwendet und in darin wird der Datentyp auch deklariert. Der EDS-Name wird dagegen im Programm angezeigt. Im Anschluss daran, können mit 'Add new Sub Index' die benötigten Strukturelemente angelegt werden. Für jeden einzelnen ist noch der Datentyp anzugeben und bei 'Name' der Elementbezeichner anzugeben. Die Elementbezeichner müssen gültige C-Variablennamen sein. Alle anderen Einträge in der Maske sind deaktiviert und können ignoriert werden. Danach kann der Datentyp bei Objekten ab Index 2000_h verwendet werden, wenn der Objektcode 'Record' ist.

5.4. Vordefinierte Symbole

Vordefinierte Präprozessorsymbole wie __DATE__, __TIME__, __FILE__ etc. können innerhalb von Strings verwendet werden. Dazu müssen diese Makros jedoch in Anführungszeichen gesetzt werden.

5.5. Spezielle Namen für Objektbeschreibungsstrukturen

Ist *Options* → *View Options* → *Use special description structures* aktiv, so wird der Name der Beschreibungsstruktur für ein Objekt nicht mehr aus dem Namen des Objekts, sondern aus dem Index abgeleitet.

5.6. Nicht-standardisierte Erweiterungen

Im Standardfall werden die COB-IDs von PDOs und SDOs nach dem Start der CANopen-Applikation nach CiA entsprechend der Node-ID und dem 'Predefined Connection Set' gesetzt. Ist dagegen *Options* → *Generate Options* → *Enable non-standard extensions* aktiv, so wird in der Datei `co_init.c` eine zusätzliche Funktion zum Überschreiben des 'Predefined Connection Set' erzeugt und die im CANopen Design Tool definierten COB-IDs werden in die EDS-Datei übernommen.

5.7. Einschränkungen

Die Optimierungen, welche im Optimierungsregister eines Index eingestellt werden, werden nur außerhalb eines Subsegments ausgeführt. Der Hintergrund ist, dass Subsegmente in Strukturen gespeichert werden und einzelnen Strukturelementen keine unterschiedliche Speicherklasse zugewiesen kann.

6. Generierung einer anwenderspezifischen Objektbeschreibung

Das Design Tool stellt Formatbeschreiber für die Generierung anwenderspezifischer Objektbeschreibungen im ASCII-Format für verschiedene Dokumententypen bereit.

Das Beschreibungstemplate ist ein Text im ASCII-Format, der Formatbeschreiber beinhalten kann für objektspezifische Informationen. Es wird in der Datei list.conf im Arbeitsverzeichnis abgespeichert. Während der Generierung aller Dateien wird das Beschreibungstemplate auf jedes Objekt angewendet. Das Design Tool speichert die Objektliste in der Datei userList.txt im Arbeitsverzeichnis.

6.1. Formatbeschreiber

Das Design Tool unterstützt folgende Formatbeschreiber:

Format-beschreiber	Beschreibung
%a	access type: RO, WO, RW, RWW, RWR, CONST
%D	refuses write on download: 0 - not valid, 1 - valid
%d	data type according to CiA-309-3 (example: u32)
%e	EDS name of the object index
%F	default value in EDS valid: 0 - not valid, 1 - valid
%g	size in bytes in decimal format (example: 12)
%I	object index in hexadecimal format without prefix (example: 1A00)
%i	object index in decimal format (example: 4096)
%L	line number in hexadecimal format without prefix (example: A)
%l	line number in decimal format (example: 12)
%M	limits in EDS valid: 0 - not valid, 1 - valid
%m	lower (minimum) limit • for numerical objects: in hexadecimal format with prefix "0x" (example: 0xFFFFFFFF) • for string objects: <empty string>
%n	name of the object index in C code
%p	PDO mapping allowed: 0 - not valid, 1 - valid
%O	refuses Read on scan: 0 - not valid, 1 - valid
%R	valid after reset: 0 - not valid, 1 - valid
%S	sub-index in hexadecimal format without prefix (example: A)
%s	sub-index in decimal format (example: 12)

Format-beschreiber	Beschreibung
%T	description of the object Take note that the Design Tool can not convert special characters because it does not know the desired document type.
%U	upper limit • for numerical objects: in hexadecimal format with prefix "0x" (example: 0xFFFFFFFF) • for string objects: <empty string>
%u	unit (example: 100 ms)
%V	default value • for numerical objects: in hexadecimal format with prefix "0x" (example: 0xFFFFFFFF) • for string objects: without quotation tags (example: Test string)
%v	default value • for numerical objects: in decimal format (example: 1614872592) • for string objects: without quotation tags (example: Test string)
%z	enumeration counter of the object in decimal format, counting starts with 0 (example: 12, i.e. it is the 12th object)

Tabelle 6, Formatbeschreiber (Beschreibung in Englisch wegen des Bezugs zur Bedienoberfläche)

6.2. Beispiel für eine HTML Dokumentation

Jedes Objekt im Objektverzeichnis wird gelistet mit Index, Sub-Index im Fettdruck und der Objektbeschreibung. Im folgenden Beispiel werden nur die Objekte 1000h und 1001h aufgeführt.

list.conf:

```
<p>
  <b> object %Ih/%S </b>: %T
</p>
```

userList.txt:

```
<p>
<b> object 1000h/0 </b>: The device type specifies the kind of device. The lower 16 bit contain the device profile number and the upper 16 bit an additional information.
</p>
<p> <b> object 1001h/0 </b>: The error register is a field of 8 bits, each for a
```

certain error type. If an error occurs the bit has to be set: Bit 0 generic error, Bit 1 current, Bit 2 voltage, Bit 3 temperature, Bit 4 communication error (overrun, error state), Bit 5 device profile specific, Bit 6 reserved, Bit 7 manufacturer specific
</p>

Anzeige im Web-Browser:

object 1000h/0: The device type specifies the kind of device. The lower 16 bit contain the device profile number and the upper 16 bit an additional information.
object 1001h/0: The error register is a field of 8 bits, each for a certain error type. If an error occurs the bit has to be set: Bit 0 generic error, Bit 1 current, Bit 2 voltage, Bit 3 temperature, Bit 4 communication error (overrun, error state), Bit 5 device profile specific, Bit 6 reserved, Bit 7 manufacturer specific

6.3. Beispiel für ein Tcl Script

Es wird eine Liste aller Objekte erzeugt. Jeder Objekteintrag beginnt mit einem Kommentar, der einen Zähler über alle Objekte enthält. Dieser Zähler wird auch als Indexer für die Objektliste objTab verwendet. Der Index, Sub-Index und der Vorbelegungswert werden für jedes Objekt gelistet. Im folgenden Beispiel werden nur die Objekte 1000h und 1001h aufgeführt.

list.conf:

```
#object %z
set objNum %z
set objTab($objNum,indexHex)      0x%I
set objTab($objNum,subHex)        %S
set objTab($objNum,defValHex)     "%V"
```

userList.txt:

```
#object 0
set objNum 0
set objTab($objNum,indexHex)      0x1000
set objTab($objNum,subHex)        0
set objTab($objNum,defValHex)     "0x00000000"
#object 1
set objNum 1
set objTab($objNum,indexHex)      0x1001
set objTab($objNum,subHex)        0
set objTab($objNum,defValHex)     "0x00"
```

Die generierte Objektliste kann nach dem Laden in einem Tcl Script benutzt werden - besonders in der bereitgestellten Testumgebung des CANopen Device Monitors.

7. CAN-Merge Erweiterungsmodul

CAN-Merge dient zur Visualisierung und Bearbeitung von Unterschieden zwischen verschiedenen CANopen Design Tool-Projekten. Das CAN-Merge Erweiterungsmodul ist ein optionales PlugIn, welches über die Lizenzdatei freigeschaltet wird. Beim Vorhandensein einer gültigen Lizenz kann CAN-Merge über das CAN-Merge-Menü in der Menüleiste des CANopen Design Tools aktiviert werden.

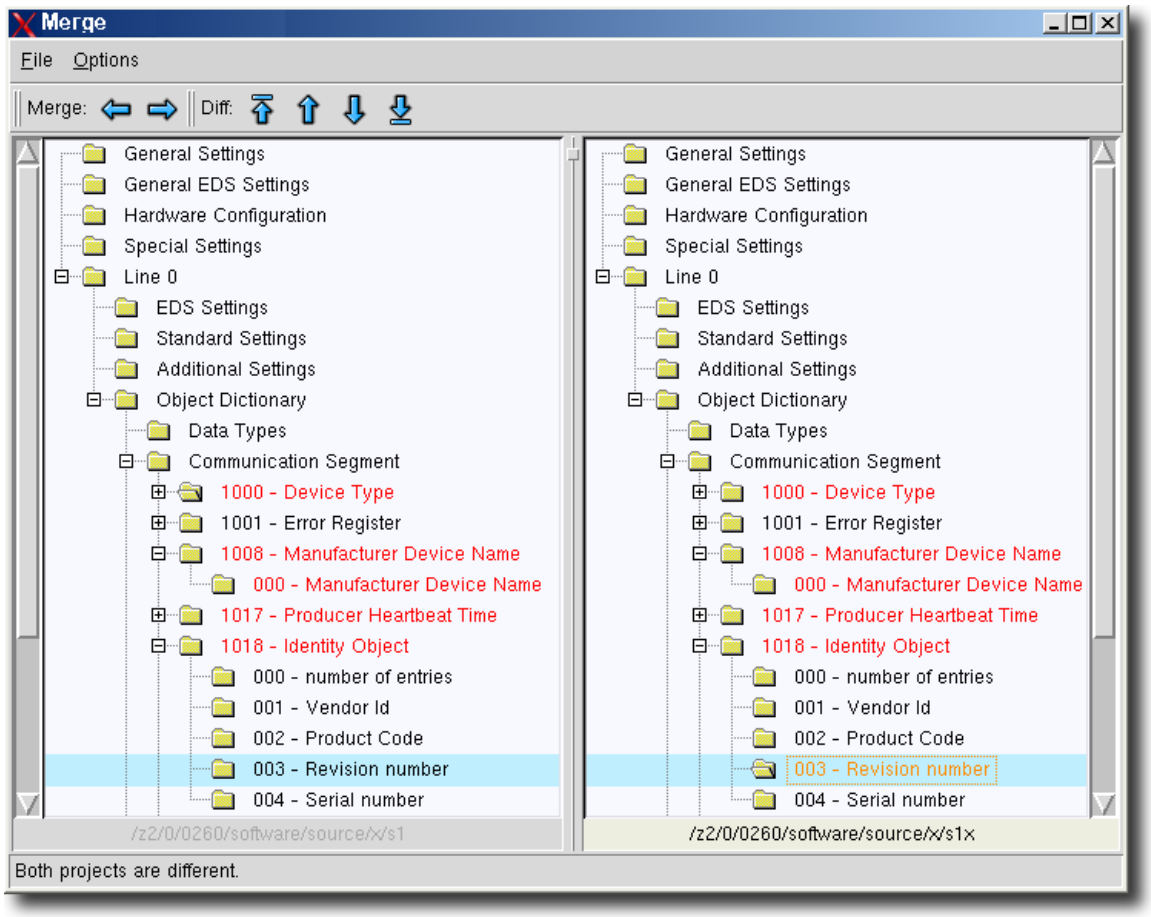


Abbildung 9, CAN-Merge-Hauptfenster mit 2 Projekten

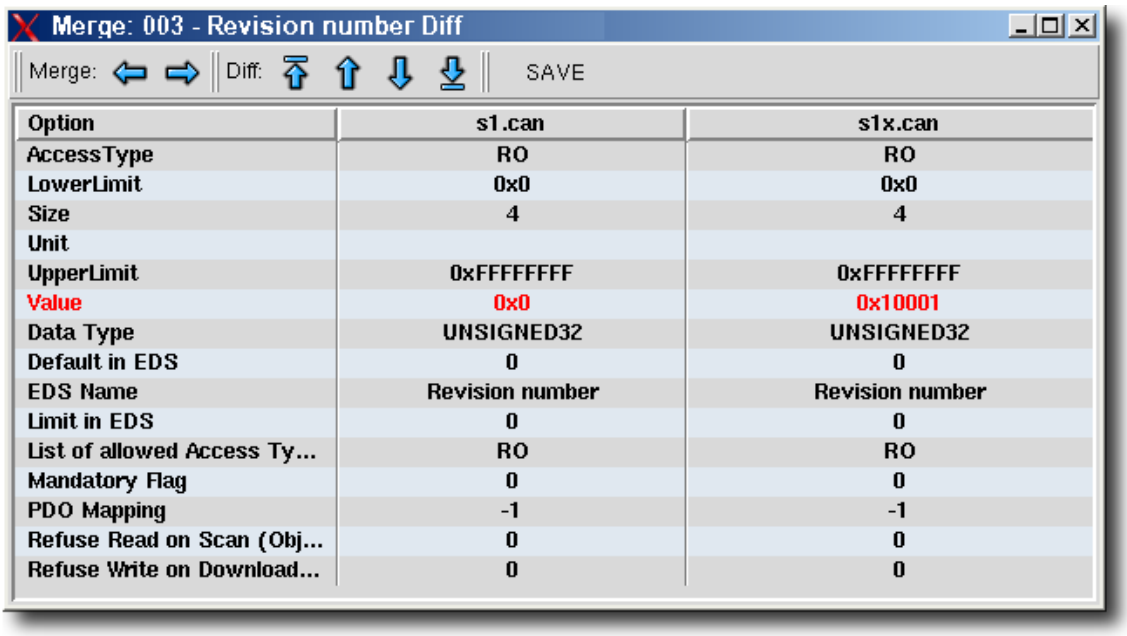
7.1. Projektansicht

In der Projektansicht werden unterschiedliche Einstellungen, Subsegmente, Objekte und Subindizes verschiedenfarbig dargestellt. Über die *Merge*-Funktion in der Werkzeugleiste können diese Elemente von einem Projekt in das andere übertragen werden.

Über das Menü *File* können die modifizierten Projekte gespeichert oder direkt in das CANopen Design Tool übernommen werden. Die Funktionen wie *Save* bzw. *Open* beziehen sich jeweils auf die aktive Projektseite. Der Menüeintrag *Generate Diff Report* in diesem Menü erzeugt eine Übersicht über die Unterschiedene der Projekte im XML-Format.

Unter *Options* lassen sich die zu vergleichenden Attribute (Diff), die zu kopierenden Attribute (Merge) und weitere Einstellungen wie das Verhalten beim Beenden von CAN-Merge und die automatische Anwahl des nächsten Unterschieds konfigurieren.

7.2. Detailansicht



Option	s1.can	s1x.can
AccessType	RO	RO
LowerLimit	0x0	0x0
Size	4	4
Unit		
UpperLimit	0xFFFFFFFF	0xFFFFFFFF
Value	0x0	0x10001
Data Type	UNSIGNED32	UNSIGNED32
Default in EDS	0	0
EDS Name	Revision number	Revision number
Limit in EDS	0	0
List of allowed Access Ty...	RO	RO
Mandatory Flag	0	0
PDO Mapping	-1	-1
Refuse Read on Scan (Obj...	0	0
Refuse Write on Download...	0	0

Abbildung 10, CAN-Merge-Difffenster mit den Unterschieden eines Subindex

Die Detailansicht stellt die Unterschiede von Einstellungen, Indizes und SubIndizes detailiert dar. Bei globalen Einstellungen ist nur die Ansicht der Unterschiede möglich. Bei Indizes und Subindizes können die Attribute direkt in der Ansicht editiert werden. Geänderte Attribute müssen explizit durch das Betätigen der *SAVE*-Schaltfläche in der Werkzeugleiste gespeichert werden. Beim Speichern wird eine Plausibilitätsprüfung (Wertebereiche, Datentypen, etc.) der Eingaben vorgenommen und die Eingabe im Fehlerfall zurückgewiesen.

8. CSV-Import Erweiterungsmodul

8.1. Anwendung

Mithilfe des CVS-Import-Erweiterungsmoduls können die Objekte des Objektverzeichnisses über CSV-Dateien importiert werden. Damit ist es möglich, Objektdefinitionen von anderen Quellen (z.B. Datenbanken) als CSV-Datei zu exportieren und danach im CANopen Design Tool zu importieren.

Das Format der CSV-Datei kann über einen *Input Description Specifier* flexibel definiert werden. Damit ist es möglich nur einen Teil der Spalten einer CSV-Datei zu verwenden und andere Einträge zu ignorieren. Zudem kann beim Import ein Offset angegeben werden.

Das CSV-Import Erweiterungsmodul ist ein optionales PlugIn, welches über die Lizenzdatei freigeschaltet wird. Beim Vorhandensein einer gültigen Lizenz kann CSV-Import-PlugIn über das CAN-Merge-Menü in der Menüleiste des CANopen Design Tools aktiviert werden.

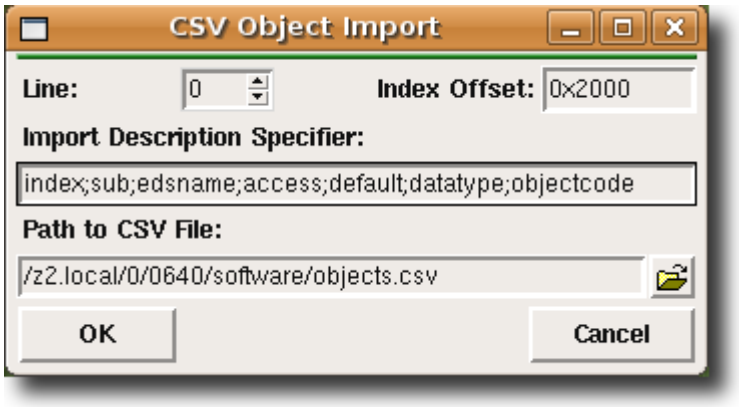


Abbildung 11, CVS-Import-Konfigurationsmaske

Nachfolgende Tabelle beschreibt alle Tokens, welche im Input Description Specifier auftreten können und ihren möglichen Wertebereich in der dazugehörigen Spalte in der CSV-Datei. Als Trenner zwischen den Spalten sind entweder ein Komma (,), ein Semikolon (;) oder ein Tabstop möglich. Es darf im Input Description Specifier nur einer der möglichen Trenner auftreten. Dieser wird dann für die CSV-Datei verwendet.

Element	Wertebereich
index	index in the range between 0x2000 and 0x9FFF
sub	sub index in the range between 0 and 254
edsname	arbitrary string
varname	valid C variable name
access	RO, RW, WO, CONST, RWR, RWW or as lower case letters
unit	arbitrary string

Element	Wertebereich
min,max,default	numbers using C notation
datatype	CANopen datatype as decimal number (see object dictionary)
objectcode	VARIABLE (default), ARRAY or SUB
desc	arbitrary string only for VARIABLE or ARRAY
createvar	flag if the c-variable shall be created (0 or 1)
pdomapping	flag if it can be mapped into a PDO (0 or 1)

8.2. Beispiele

Das Beispiel zeigt eine Standardkonfiguration für den Import von herstellerspezifischen Objekten.

Input Description Specifier:

```
index,sub,varname,edsname,unit,access,min,max,default,datatype, \
objectcode,createvar,pdomapping
```

CSV File:

```
0x3000,0,man_eins,EDSName_1,,ro,0,0xff,0xF0,5,VARIABLE,1,0
0x3001,0,man_zwei,Variable_2,,ro,0,0xff,0xF0,5,VARIABLE,1,0
0x3002,0,man_drei,Ohne_Create_Var,,ro,0,0xff,0xF0,5,VARIABLE,0,1
0x4000,,min_array,Lufthupen,,rw,,,,6,ARRAY,0,0
0x4000,1,,Lufthupe_1,,wo,0,0xf1,0x01,6,SUB,1,1
0x4000,2,,Lufthupe_2,,ro,0,0xf2,0x02,6,SUB,1,1
0x4000,3,,Lufthupe_3,,rwr,0,0xf3,0x03,6,SUB,1,1
```

Im nächsten Beispiel werden einige Spalten aus der CSV-Datei ignoriert.

Input Description Specifier:

```
;;;;index;;;varname;;;access;default;datatype;objectcode;edsname
```

CSV File:

```
whatever;;ign;;0x2001;0;0;test_1;;;ro;0;6;VARIABLE;Number of Inputs
whatever;;ign;;0x2002;0;0;test_2;;;ro;0;6;VARIABLE;Number of Outputs
whatever;;ign;;0x2003;0;0;test_3;;;ro;0;6;VARIABLE;Number of Variables
```

9. Unterschiedliche Versionen

9.1. Standardversion

Die Standardversion beinhaltet die in den vorigen Kapiteln beschriebenen Funktionen ohne Einschränkungen.

9.2. Light Version

In der Light-Version ist bei der Generierung die Anzahl der Objekte auf 15 begrenzt. Ist die Anzahl der Objekte größer als 15, so werden die Ausgabedateien nicht generiert.

9.2.1. Einschränkungen

Ist die Anzahl der Objekte kleiner als 15, so werden zu Demonstrationszwecken alle Ausgaben erzeugt. Wird das Limit jedoch überschritten, so kann die 'Light'-Version nur die Bibliothekskonfigurationsdatei **cal_conf.h** erzeugen.

9.2.2. Anwendung

Wenn keine spätere Verwendung der Projektdateien in einer Standardversion des CANopen Design Tools angedacht ist, so müssen nur folgende Masken und Eingabefelder berücksichtigt werden:

- General Settings
- Hardware Configuration

Zur Konfiguration der CANopen Library entsprechend den verwendeten Dienste ist folgende Vorgehensweise empfehlenswert:

- Anlegen der gewünschten Anzahl an Linien
- In jeder Linie ist das Communication Segment vollständig auszufüllen.

Ist die Konfiguration abgeschlossen, so kann mit *Generate all Files* die Datei **cal_conf.h** erstellt werden. Die Projektdatei des CANopen Design Tools light ist jedoch zu sichern, da aus einer **cal_conf.h** kein Import der Bibliothekskonfiguration möglich ist.
Die Projektdatei kann mit allen anderen Varianten des CANopen Design Tools weiterverwendet werden.

9.3. CANopen EDS Editor

9.3.1. Einschränkungen

Die Generierung der Programmausgaben ist auf die Dokumentation und die XDD/EDS-Dateien beschränkt.

9.3.2. Anwendung

Der CANopen EDS Editor ist eine reduzierte Version des CANopen Design Tools. Daher besteht die Möglichkeit CANopen Design Tool-Projektdateien auch für die Bearbeitung und Speicherung von EDS/XDD-Dateien zu verwenden. Alternativ dazu können EDS/XDD-Dateien auch direkt über "File → Import EDS/XDD file" geladen und über "Generate → Generate all files" generiert werden.

Wenn keine spätere Verwendung der Projektdateien in einer Standardversion des CANopen Design Tools angedacht ist, so müssen nur folgende Masken und Eingabefelder berücksichtigt werden:

- General EDS/XDD Settings
  - CharacteristicsList
- EDS/XDD Settings (line specific)
- In der Strukturansicht der Indizes:
  - EDS Name
  - Data Type
  - Object Code
  - Description (nur für die XDD-Paramterbeschreibung und die Dokumentation)
- In der Strukturansicht der Subindizes:
  - EDS Name (bei Arrays und Records)
  - Data Type
  - Access
  - PDO Mapping
  - Value
  - Unit (nur für die Dokumentation)

10. Konvertierung bestehender Projekte

10.1. Bestehende Projekte - Version >= 2.2

Vorhandene Projekte ab dieser Version können einfach über die vorhandene .can-Datei geöffnet werden, wobei alle Einstellungen übernommen werden. Über relevante Neuerungen bei Hardware- oder Projekteinstellungen, welche seit der Version des Projekts hinzugefügt wurden, werden Sie beim Öffnen der Projektdatei informiert.

10.2. Bestehende Projekte - Version >= 2.0

Zum Import von bestehenden Projekten der Versionen 2.0 oder 2.1 öffnen Sie einfach die vorhandene .can-Datei.

Folgende Punkte müssen dennoch beachtet werden:

- Informationen über "Object Dictionary Optimizations" werden nicht übernommen
- Die SYNC-Objekte (1005_h etc.) müssen ggf. gelöscht und erneut eingefügt werden.
- Prüfen Sie bitte alle Hardware-Einstellungen! Bei mehreren Hardware-Plattformen wurden die Konfigurationsmöglichkeiten erweitert und müssen entsprechend konfiguriert werden.
- Nach dem Speichern mit dieser Version des CANopen Design Tools kann die Projektdatei nicht mehr mit einer früheren Version geöffnet werden.

10.3. Bestehende Projekte - Version < 2.0

Existierende Projekte, welche konvertiert werden können, bestehen aus einem CANopen Design Tool Projekt der Versionen 1.5 bis 1.7.2 und einer **cal_conf.h** Datei der CANopen Library Version 4.3.
Aus dem CANopen Design Tool Projekt werden die Objektverzeichnisse und die EDS-Einstellungen extrahiert und die Hardware- und Bibliothekskonfiguration wird aus der Datei **cal_conf.h** extrahiert.

10.3.1. Das Konvertierungstool

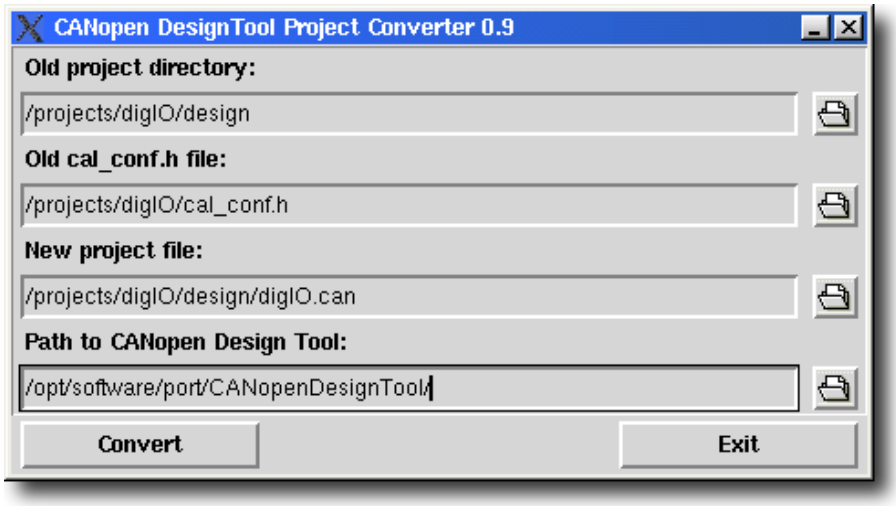


Abbildung 12 , Konvertierungstool

Das Konvertierungstool **dtConv.exe** (bzw. dtConv unter Linux) kann ohne Installation direkt gestartet werden, wonach die oben dargestellte Maske erscheint.

Sind die Pfade[3] in den jeweiligen Feldern korrekt eingestellt, so kann die Konvertierung mit dem Betätigen der 'Convert'-Schaltfläche gestartet werden. Ist der Pfad zum aktuellen CANopen Design Tool nicht korrekt oder ist das aktuelle CANopen Design Tool nicht vorhanden, so können die Projekte nicht konvertiert werden.

Vor der eigentlichen Konvertierung werden die Profile des neuen CANopen Design Tool eingelesen und es muss angegeben werden welche Profile im bestehenden Projekt verwendet wurden.

Abbildung 13 , Profilauswahldialog

Während der Konvertierung öffnet sich ein Nachrichtenfenster in dem Statusmeldungen, Hinweise und Warnungen ausgegeben werden.

[3] Der Pfad des alten CANopen Design Tool-Projekts ist das Verzeichnis, welches das *line0*-Verzeichnis enthält.

10.3.2. Einschränkungen

10.3.2.1. Segmentierung des Objektverzeichnisses

Informationen über die Segmentierung des Objektverzeichnisses gehen bei der Konvertierung verloren. Die Objekte innerhalb eines Segments werden dennoch richtig konvertiert und die Segmentinformationen können im CANopen Design Tool nachgetragen werden.

10.3.2.2. Benutzerdefinierte Datentypen

Benutzerdefinierte Datentypen werden u.U. nicht richtig konvertiert. Jedoch wird jede Verwendung eines unbekannten Datentyps erkannt und im Nachrichtenfenster mit der Linienummer und dem Index ausgegeben, so dass die Indizes überarbeitet werden.
Dabei ist zu beachten, dass sich das Konzept der Definition benutzerdefinierter Datentypen gegenüber früheren Versionen geändert hat. (Siehe Abschnitt *Herstellerspezifische Strukturen* in Kapitel 5)

Im Allgemeinen ist es weiterhin angebracht alle konvertierten Einstellungen sorgfältig zu prüfen.

10.4. Konvertierungssupport

Falls bei der Konvertierung Probleme oder Unregelmäßigkeiten auftreten, so setzen Sie sich bitte mit uns in Verbindung.

Anhang 1 — Abkürzungen

CAN	Controller Area Network
CAL	CAN Application Layer (CANopen base)
CiA	CAN in Automation international users and manufacturers group e.V.
COB	Communication Object (CAN Message)
COB-ID	Communication Object Identifier
DDE	Device Description Entry
EMCY	Emergency Object
EPL	Ethernet POWERLINK
EPSG	Ethernet POWERLINK Standardization Group
ESI	EtherCAT Slave Information XML-Gerätebeschreibung für EtherCAT Slaves
ETG	EtherCAT Technology Group
NMT	Network Management
PDO	Prozeß-Daten-Objekt, unbestätigter Dienst für die Echtzeitkommunikation
RPDO	Receive PDO (Empfangs-PDO)
SDO	Service-Daten-Objekt, bestätigter Dienst für den Zugriff auf das Objektverzeichnis
CSDO	Client SDO
SSDO	Server SDO
SYNC	Sychronization Object
TIME	Time Stamp Object (Zeitstempel)
TPDO	Transmit PDO (Sende-PDO)
XDD	XML Device Description - Gerätebeschreibungsdateien für POWERLINK oder CANopen
XML	eXtensible Markup Language